UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/812,619	03/20/2001	William Jon Schmidt	ROC9-2000-0281-US1	1322

46296 7590 04/21/2005

MARTIN & ASSOCIATES, LLC
IBM INTELLECTUAL PROPERTY LAW DEPARTMENT
DEPARTMENT 917, BUILDING 006-1
3605 HIGHWAY 52 NORTH
ROCHESTER, MN 55901-7829

EXAMINER	
WOOD, WILLIAM H	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 04/21/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/812,619	SCHMIDT, WILLIAM JON	
	Examiner	Art Unit	
	William H. Wood	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 October 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-8,10,13-15 and 18-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,4-8,10,13-15 and 18-23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claims 1, 4-8, 10, 13-15 and 18-23 are pending and have been examined.

Double Patenting

1. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

2. Claims 1-20 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1, 7, 9 and 12-14 of U.S. Patent No. 6,505,344 (**Blais** et al.) in view of **Whaley** et al., "Compositional Pointer and Escape Analysis for Java Programs". Although the conflicting claims are not identical, they are not patentably distinct from each other because the instant claims recite an obvious variation of copending application (now patented). In regard to claim 1 (of instant application), **Blais** disclosed an apparatus comprising: at least one processor; a memory coupled to the at least one processor; an object oriented program residing in the memory comprising a plurality of instructions; and compiler residing in the memory and executed by the at least one processor. **Blais** did not explicitly disclose a dynamic

compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object. **Whaley** demonstrated that it was known at the time of invention to utilize dynamic compilers in escape analysis (page 188, left column, first paragraph above section 1.3) and allocate objects to the stack if they do not escape, "NoEscape", from the calling method (page 201-202, section 7.2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the stack allocation system of **Blais** with allocation to the stack for objects that don't escape a method and dynamic compilation as found in **Whaley**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide escape analysis for memory optimization (and thus improving efficiency in memory usage) to **Blais**, which is already performing escape analysis. Additionally, providing dynamic compilation is motivated by providing the disclosed features to as many compiling environments as possible to improve code development systems. Other independent claims are related to claim 1.

3. Claim 1-20 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1, 7, 9 and 12-14 of copending Application No. 09/865,001 (**Blais** et al.) in view of **Whaley** et al., "Compositional Pointer and Escape Analysis for Java Programs". Although the conflicting claims are not identical, they are not patentably distinct from each other because the instant claims recite an obvious variation of copending application. In regard to claim 1 (of instant application), **Blais** disclosed an apparatus comprising: at

least one processor; a memory coupled to the at least one processor; an object oriented program residing in the memory comprising a plurality of instructions; and compiler residing in the memory and executed by the at least one processor. **Blais** did not explicitly disclose a dynamic compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object. **Whaley** demonstrated that it was known at the time of invention to utilize dynamic compilers in escape analysis (page 188, left column, first paragraph above section 1.3) and allocate objects to the stack if they do not escape, "NoEscape", from the calling method (page 201-202, section 7.2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the stack allocation system of **Blais** with allocation to the stack for objects that don't escape a method and dynamic compilation as found in **Whaley**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide escape analysis for memory optimization (and thus improving efficiency in memory usage, stack verses heap allocation) to **Blais**. Additionally, providing dynamic compilation is motivated by providing the disclosed features to as many compiling environments as possible to improve code development systems. Other independent claims are related to claim 1.

This is a provisional obviousness-type double patenting rejection.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 13-15, 18-20, 21-23 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Signal bearing media of independent claims 13, 18 and 21 are not tangible as evidenced by Applicant's Specification page 33, lines 18-20. Tangible forms of media include floppy disks. "Transmission" media or broadcast media are not considered tangible by the office and thus not patentable.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 4, 5, 10 and 21-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Whaley** et al., "Compositional Pointer and Escape Analysis for Java Programs" in view of **Hölzle** et al. (USPN 6,237,141).

Claim 4

Whaley disclosed an apparatus comprising:

- at least one processor *(abstract; processor for executing Java programs)*;

- a memory coupled to the at least one processor *(abstract; memory for storing Java programs)*;
- an object oriented program residing in the memory comprising a plurality of instructions *(abstract; Java programs)*; and
- a dynamic compiler residing in the memory and executed by the at least one processor *(page 188, left column, first paragraph above section 1.3)*, the dynamic compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object *(page 201-202, section 7.2)*.

Whaley did not explicitly state invoking dynamic compiler during execution of object oriented program. **Hölzle** demonstrated that it was known at the time of invention to provide an execution system, which interprets and determines when dynamic compilation is needed (column 3, lines 15-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiler optimization analysis techniques of **Whaley** with invoking dynamic compilation for parts of a program as found in **Hölzle**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make intelligent decisions about when to interpret and when to compile (column 3, lines 22-25), both interpretation and compilation being of value in an efficient system (column 1, lines 44-53).

Whaley did not explicitly state the apparatus wherein the dynamic compiler *changes* the allocation of the object from the invocation stack frame to a heap due to information that

becomes available from at least one class that is loaded after the dynamic compiler allocates the at least one object to the invocation stack frame. **Whaley** demonstrated that it was known at the time of invention: Java allocates all objects to the heap (page 187, section 1.1, first sentence); incremental analysis and dynamic compilation (page 188, right column everything above section 1.3); and allocating to the stack allows for "automatic" garbage collection (pages 201-202, sentence: "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the Java analysis system of **Whaley** with changing object allocation to the heap as more information becomes available (when it is determined to escape) as suggested by **Whaley**'s own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to not automatically "garbage collect" an object whose lifetime continues beyond a particular method and thus avoiding possibly loosing information or damaging the normal operation of the program.

Claim 5

Whaley did not explicitly state the apparatus of claim 4 wherein the dynamic compiler changes at least one pointer to the object allocated on the invocation stack to point to an object allocated on the heap as a result of information that becomes available as more classes that are part of the object oriented program are loaded. **Whaley** demonstrated that it was known at the time of invention: Java allocates all objects to

the heap (page 187, section 1.1, first sentence); incremental analysis and dynamic compilation (page 188, right column everything above section 1.3); and allocating to the stack allows for "automatic" garbage collection (pages 201-202, sentence: "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the Java analysis system of **Whaley** with changing object allocation to the heap as more information becomes available (when it is determined to escape) and thus pointers to the objects as suggested by **Whaley**'s own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to not automatically "garbage collect" an object whose lifetime continues beyond a particular method and thus avoiding possibly loosing information or damaging the normal operation of the program.

Claim 10

Whaley disclosed a method for allocating objects in an object oriented program to memory *(page 187, section 1.1)*, the method comprising:

- loading a plurality of classes that are part of the object oriented program *(page 188, first paragraph above section 1.3; and page 202, section 8.1)*;
- executing code from at least one of the plurality of loaded classes *(page 202, section 8.1)*;
- allocating at least one object to an invocation stack frame for a method that allocates the at least one object *(page 201-202, section 7.2)*.

Whaley did not explicitly state *determining* whether dynamic compilation of a portion of the object oriented program is needed. **Hölzle** demonstrated that it was known at the time of invention to provide an execution system, which interprets and determines when dynamic compilation is needed (column 3, lines 15-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiler optimization analysis techniques of **Whaley** with determining dynamic compilation for parts of a program as found in **Hölzle**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make intelligent decisions about when to interpret and when to compile (column 3, lines 22-25), both interpretation and compilation being of value in an efficient system (column 1, lines 44-53).

Whaley did not explicitly state the apparatus wherein the dynamic compiler *changes* the allocation of the object from the invocation stack frame to a heap due to analysis of subsequently loaded classes. **Whaley** demonstrated that it was known at the time of invention: Java allocates all objects to the heap (page 187, section 1.1, first sentence); incremental analysis and dynamic compilation (page 188, right column everything above section 1.3); and allocating to the stack allows for "automatic" garbage collection (pages 201-202, sentence: "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the Java analysis system of **Whaley** with changing object allocation to the heap as more

information becomes available (when it is determined to escape) as suggested by **Whaley**'s own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to not automatically "garbage collect" an object whose lifetime continues beyond a particular method and thus avoiding possibly loosing information or damaging the normal operation of the program.

Claims 21-23

The limitations of claim 21 correspond to claim 4 and as such are rejected in the same manner. Transmission and recordable signal bearing media are inherent to the disclosed system of **Whaley** and **Hölzle**.

8. Claims 1, 6-8, 13-15, 18 and 19-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Whaley** et al., "Compositional Pointer and Escape Analysis for Java Programs" in view of **Hölzle** et al. (USPN 6,237,141) and in further view of **Choi** et al., "Escape Analysis for Java".

Claim 1

Whaley disclosed an apparatus comprising:

- at least one processor *(abstract; processor for executing Java programs)*;
- a memory coupled to the at least one processor *(abstract; memory for storing Java programs)*;

- an object oriented program residing in the memory comprising a plurality of instructions *(abstract; Java programs)*; and
- a dynamic compiler residing in the memory and executed by the at least one processor *(page 188, left column, first paragraph above section 1.3)*, the dynamic compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object *(page 201-202, section 7.2)*.

Whaley did not explicitly state invoking dynamic compiler during execution of object oriented program. **Hölzle** demonstrated that it was known at the time of invention to provide an execution system, which interprets and determines when dynamic compilation is needed (column 3, lines 15-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiler optimization analysis techniques of **Whaley** with invoking dynamic compilation for parts of a program as found in **Hölzle**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make intelligent decisions about when to interpret and when to compile (column 3, lines 22-25), both interpretation and compilation being of value in an efficient system (column 1, lines 44-53).

Whaley disclosed the apparatus wherein the dynamic compiler comprises:

an escape analysis mechanism that marks each instruction that allocates a new object based on information available from previously-loaded classes that are part of the

object oriented program *(page 188, left column above section 1.3, analysis becoming more precise)*; and

an object allocation mechanism that allocates at least one object to an invocation stack frame for a method that allocates the object *(page 201-202, section 7.2)*.

Whaley did not explicitly state the markings of global, no and arg escape and marking "no escape". **Choi** demonstrated that it was known at the time of invention to perform escape analysis using such markings (page 2-3, sections 2 and 2.1, specifically "proposition 2.3" and down). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the escape analysis of **Whaley** with such markings as found in **Choi**'s teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make use of a simple and efficient data flow mechanism (**Choi**: page 1, Abstract).

Whaley did not explicitly state the apparatus of claim 2 wherein the dynamic compiler analyses each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object by the object allocation mechanism to the invocation stack frame, and if so, the dynamic compiler changes the allocation of the object to the heap. **Whaley** demonstrated that it was known at the time of invention: Java allocates all objects to the heap (page 187, section 1.1, first sentence); incremental analysis and dynamic compilation (page 188, right column everything above section 1.3; page 202, first paragraph of section 8.1); and allocating to the stack allows for "automatic" garbage collection (pages 201-202, sentence: "Instead of being processed by the collector, the

object will be implicitly collected when the method returns and the stack rolls back"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the Java analysis system of **Whaley** with changing object allocation to the heap as more information becomes available (when it is determined to escape) as suggested by **Whaley**'s own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to not automatically "garbage collect" an object whose lifetime continues beyond a particular method and thus avoiding possibly loosing information or damaging the normal operation of the program.

Claim 6

Whaley disclosed an apparatus comprising:

- at least one processor *(abstract; processor for executing Java programs)*;
- a memory coupled to the at least one processor *(abstract; memory for storing Java programs)*;
- an object oriented program residing in the memory comprising a plurality of instructions *(abstract; Java programs)*; and
- a dynamic compiler residing in the memory and executed by the at least one processor *(page 188, left column, first paragraph above section 1.3)*, the dynamic compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object *(page 201-202, section 7.2)*.

Whaley did not explicitly state invoking dynamic compiler during execution of object oriented program. **Hölzle** demonstrated that it was known at the time of invention to provide an execution system, which interprets and determines when dynamic compilation is needed (column 3, lines 15-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiler optimization analysis techniques of **Whaley** with invoking dynamic compilation for parts of a program as found in **Hölzle**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make intelligent decisions about when to interpret and when to compile (column 3, lines 22-25), both interpretation and compilation being of value in an efficient system (column 1, lines 44-53).

Whaley disclosed the apparatus wherein the dynamic compiler comprises:

an escape analysis mechanism that marks each instruction that allocates a new object based on information available from previously-loaded classes that are part of the object oriented program *(page 188, left column above section 1.3, analysis becoming more precise)*; and

an object allocation mechanism that allocates at least one object to an invocation stack frame for a method that allocates the object *(page 201-202, section 7.2)*.

Whaley did not explicitly state the markings of global, no and arg escape and marking "no escape". **Choi** demonstrated that it was known at the time of invention to perform escape analysis using such markings (page 2-3, sections 2 and 2.1, specifically

"proposition 2.3" and down). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the escape analysis of **Whaley** with such markings as found in **Choi**'s teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make use of a simple and efficient data flow mechanism (**Choi**: page 1, Abstract).

Whaley did not explicitly state the apparatus wherein the dynamic compiler *changes* the allocation of the object from the invocation stack frame to a heap due to information that becomes available from at least one class that is loaded after the dynamic compiler allocates the at least one object to the invocation stack frame. **Whaley** demonstrated that it was known at the time of invention: Java allocates all objects to the heap (page 187, section 1.1, first sentence); incremental analysis and dynamic compilation (page 188, right column everything above section 1.3); and allocating to the stack allows for "automatic" garbage collection (pages 201-202, sentence: "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the Java analysis system of **Whaley** with changing object allocation to the heap as more information becomes available (when it is determined to escape) as suggested by **Whaley**'s own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to not automatically "garbage collect" an object whose lifetime continues beyond a particular

method and thus avoiding possibly loosing information or damaging the normal operation of the program.

Claim 7

The limitations of method claim 7 correspond to the limitations of claim 1 and as such are rejected in the same manner.

Claim 8

Whaley and **Hölzle** disclosed the method of claim 7 wherein act (A) comprises determining whether a method in the portion has been executed a number of times equal to or greater than a predetermined threshold value *(**Hölzle**: column 3, lines 20-22)*.

Claim 13

The limitations of claim 13 correspond to the limitations of apparatus claim 7 and as such are rejected in the same manner. Signal bearing media inherent to programmable computer implementation.

Claim 14

Whaley and **Hölzle** disclosed the program product of claim 13 wherein the signal bearing media comprises recordable media *(inherent to programmable computer implementation)*.

Claim 15

Whaley and **Hölzle** disclosed the program product of claim 13 wherein the signal bearing media comprises transmission media *(inherent to programmable computer implementation)*.

Claim 18

The limitations of claim 18 correspond to the limitations of apparatus claim 6 and as such are rejected in the same manner.

Claim 19-20

The limitations of claims 19-20 correspond to the limitations of apparatus claims 14 and 15 and as such are rejected in the same manner.

Response to Arguments

9. Applicant's arguments filed 04 October 2004 have been fully considered but they are not persuasive. Applicant argued citations of prior art **Whaley** fail to demonstrate Applicant's claimed invention. It is respectfully disagreed. First, the cited portions of **Whaley** are evidence supporting a rejection under U.S.C 103 obviousness. As such, the cited portions do not explicitly state *changing an allocation from stack to heap.* The cited prior art makes clear allocation is normally performed to the heap, their escape analysis is dynamic and iterative (**Whaley**: page 188, left column first two paragraphs),

and the stack is used only if it doesn't escape. Thus, in such an iterative system if an escape occurs the allocation should obviously be moved (changed from stack) to the heap. Therefore, the rejections are maintained using the prior art of **Whaley**.

Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to William H. Wood whose telephone number is (703)305-3305. The examiner can normally be reached 7:30am - 5:00pm Monday thru Thursday and 7:30am - 4:00pm every other Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662. The fax phone numbers for the organization where this application or proceeding is assigned are (703)746-7239 for regular communications and (703)746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703)305-3900.

William H. Wood
April 14, 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100